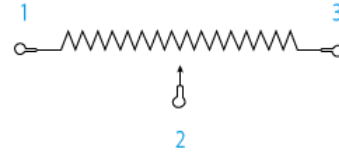


4. 센서활용-2

이영주
young.kopo@gmail.com

VR 상태 모니터링(통신, I2C)



가변저항(Potentiometer)은 사용자가 직접 저항 값을 임의로 바꿀 수 있는 저항기



멀티턴
가변저항



싱글터
가변저항

패널장착용 포텐셔미터



트림머 포텐셔미터

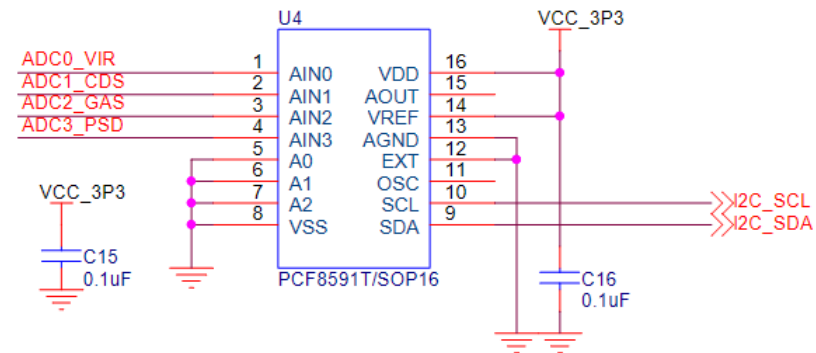


슬라이드식 가변저항



360° 연속회전 가변저항
정밀 포텐셔미터

ADC 제어(통신, I2C)



The PCF8591 is a single-chip, single-supply low-power 8-bit CMOS data acquisition device with four analog inputs, one analog output and a serial I²C-bus interface. Three address pins A0, A1 and A2 are used for programming the hardware address, allowing the use of up to eight devices connected to the I²C-bus without additional hardware. Address, control and data to and from the device are transferred serially via the two-line bidirectional I²C-bus.

The functions of the device include analog input multiplexing, on-chip track and hold function, 8-bit analog-to-digital conversion and an 8-bit digital-to-analog conversion. The maximum conversion rate is given by the maximum speed of the I²C-bus.

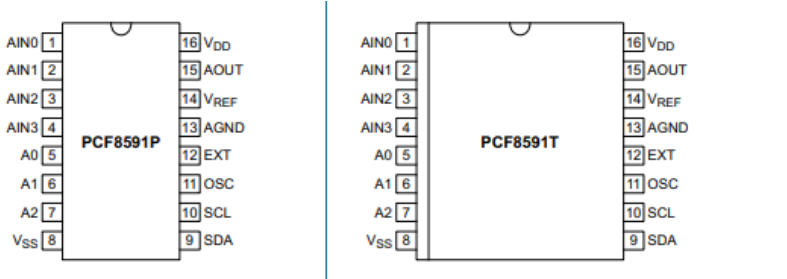


Table 5. I²C slave address byte

Bit	Slave address							0
	7	6	5	4	3	2	1	
	MSB				A2	A1	A0	LSB
slave address	1	0	0	1	A2	A1	A0	R/W

The least significant bit of the slave address byte is bit R/W (see Table 6).

PCF8591 I2C Address's

The lower three bits of the address consist of the three digital inputs A2, A1, A0 while the upper bits are fixed at 1001xxx. The the last bit(LSB 'L') is ignored as it is the read write bit (R/Wn). Therefore the addresses available are:

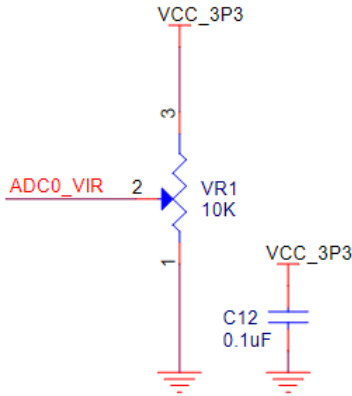
0x48, 0x49, 0x4A, 0x4B, 0x4C, 0x4D, 0x4E, 0x4F.

There are eight individual addresses selected by setting the inputs A2..A0 high or low in hardware.

Pin description

Table 4. Pin description

Symbol	Pin	Description
AIN0	1	analog inputs (A/D converter)
AIN1	2	
AIN2	3	
AIN3	4	
A0	5	hardware slave address
A1	6	
A2	7	
VSS	8	ground supply voltage
SDA	9	I ² C-bus serial data input and output
SCL	10	I ² C-bus serial clock input
OSC	11	oscillator input/output
EXT	12	external/internal switch for oscillator input
AGND	13	analog ground supply
VREF	14	voltage reference input
AOUT	15	analog output (D/A converter)
VDD	16	supply voltage



ADC 제어(통신, I2C) - 파이썬

파이썬 통신 모듈:

```
import smbus
```

```
bus = smbus.SMBus(1)
```

```
# I2C버스가 연결된 smbus 모듈 객체화
```

```
i2c_address = 0x48
```

```
command = 0x44 (데이터 읽어올때)
```

```
# I2C 사용을 위한 모듈 smbus
```

```
import smbus
```

```
# 지연시간 제어를 위해 time 모듈을 사용한다.
```

```
import time
```

```
bus = smbus.SMBus(1)
```

```
i2c_address = 0x48
```

```
# PCF8591 칩에서 데이터를 받기위한 명령어다.
```

```
command = 0x44
```

```
# try-except 는 파이썬의 예외처리 구문으로
```

```
# 키보드로 Ctrl + C를 누를시 프로그램이 종료 된다.
```

```
try:
```

```
    while(1):
```

```
        # i2c의 주소와 명령어를 전송하여 5Byte의 데이터를 읽어온다.
```

```
        # 맨 앞의 dummy data(index 0번 데이터)를 제외하고 뒤 4Byte가 4개의 ADC 포트 데이터이다.
```

```
        adc_data = bus.read_i2c_block_data(i2c_address, command, 5)
```

```
        # I2C로 부터 센서 값을 읽어온다 VR 센서는 ADC 0번이다.
```

```
        VrValue = adc_data[1]
```

```
        VrValue = VrValue * 100 / 255
```

```
        VrValue = round(VrValue, 2)
```

```
        print("가변저항 : " + str(VrValue) + " %")
```

```
        time.sleep(0.1)
```

```
# 종료 등의 키보드 인터럽트 발생시 처리 동작
```

```
except KeyboardInterrupt:
```

```
    pass
```



```
28          VrValue = round(VrValue, 2)
29
Shell
가변저항 : 62.35 %
가변저항 : 62.35 %
가변저항 : 62.35 %
가변저항 : 62.35 %
가변저항 : 62.35 %
가변저항 : 62.35 %
가변저항 : 62.35 %
```

ADC 제어(통신, I2C) - WiringPi

```
#include <stdio.h>
#include <wiringPi.h>
#include <wiringPiI2C.h>
int main(void) {
    int fd;
    int prev, a2dVal[4];
    float a2dVol;
    float Vref = 5.0;
    printf("[ADC/DAC Module testing.....]\n");
    if ((fd = wiringPiI2CSetup(0x48)) < 0) {
        printf("wiringPiI2CSetup failed:\n");
    }
    while (1) {
        wiringPiI2CWrite(fd, 0x44);
        prev = wiringPiI2CRead(fd); // Previously byte, garbage
        for(int i = 0; i < 4; i++) a2dVal[i] = wiringPiI2CRead(fd);
        a2dVol = a2dVal[0] * 100.0 / 255;
        printf("VAR = %3.0f %n", a2dVol);
        delay(500);
    }
}
```

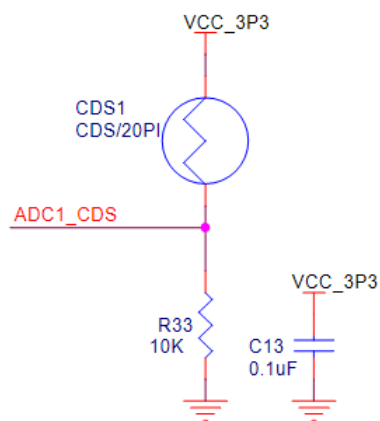


```
bready@raspberrypi:~/AISW $ ./adc_auto
[ADC/DAC Module testing.....]
VAR = 62 %
VAR = 62 %
VAR = 62 %
VAR = 62 %
VAR = 62 %
VAR = 62 %
VAR = 62 %
VAR = 62 %
VAR = 62 %
VAR = 62 %
VAR = 62 %
```

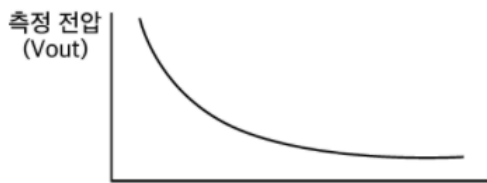
조도센서 CDS 제어(통신, I2C)



- 조도 센서(Photo Resistor)는 주변 환경의 밝기를 측정할 수 있는 센서
- 광에너지(빛)를 받으면 내부에 움직이는 전자가 발생하여 전도율이 변하는 광전효과를 가지는 소자



- Cds 센서는 옥외주차장, 휴식중의 극장 객석 정도의 빛의 양(10 Lux)에서 약 10K Ω 의 저항을 가지며, 아무런 빛이 없는 암실에서 약 200K Ω 의 저항
- Cds 센서가 저항이기 때문에, 빛의 양이 매우 많은 경우, 저항수치가 매우 작아져 과전류가 흐를 수 있다.
- 조도를 측정하기 위해서는 풀업 회로를 사용하며 풀업 저항 사용시에는 조도 센서의 저항 값에 따라 전압 분배가 일어나 주변이 밝을수록 측정되는 전압의 크기는 작아진다.



ADC 제어 조도센서 (통신, I2C) - 파이썬

파이썬 통신 모듈:

```
import smbus
```

```
bus = smbus.SMBus(1)
```

```
# I2C버스가 연결된 smbus 모듈 객체화
```

```
i2c_address = 0x48
```

```
command = 0x44 (데이터 읽어올때)
```

```
# I2C 사용을 위한 모듈 smbus
```

```
import smbus
```

```
# 지연시간 제어를 위해 time 모듈을 사용한다.
```

```
import time
```

```
bus = smbus.SMBus(1)
```

```
i2c_address = 0x48
```

```
# PCF8591 칩에서 데이터를 받기위한 명령어다.
```

```
command = 0x44
```

```
# try-except 는 파이썬의 예외처리 구문으로
```

```
# 키보드로 Ctrl + C를 누를시 프로그램이 종료 된다.
```

```
try:
```

```
    while(1) :
```

```
        # i2c의 주소와 명령어를 전송하여 5Byte의 데이터를 읽어온다.
```

```
        # 맨 앞의 dummy data(index 0번 데이터)를 제외하고 뒤 4Byte가 4개의 ADC 포트 데이터이다.
```

```
        adc_data = bus.read_i2c_block_data(i2c_address, command, 5)
```

```
        # I2C로 부터 센서 값을 읽어온다 CdS 센서는 1번이다.
```

```
        CdsValue = adc_data[2]
```

```
        CdsValue = CdsValue * 100 / 255
```

```
        # 소수점 둘 째 자리까지만 표시한다
```

```
        CdsValue = round(CdsValue,2)
```

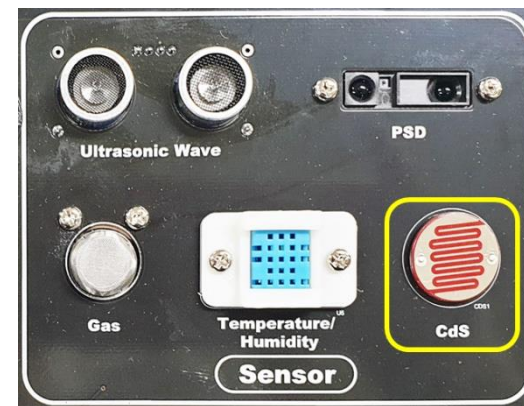
```
        print("CdS : " + str(CdsValue) + " %")
```

```
        time.sleep(0.1)
```

```
# 종료 등의 키보드 인터럽트 발생시 처리 동작
```

```
except KeyboardInterrupt:
```

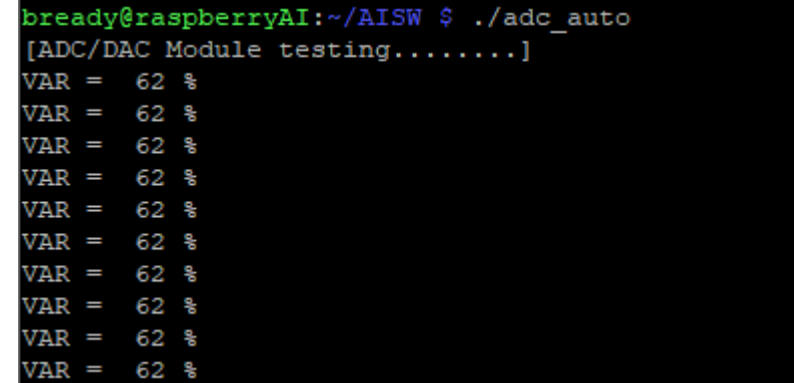
```
    pass
```



```
CdS : 84.71 %  
CdS : 85.88 %  
CdS : 87.06 %  
CdS : 86.27 %
```

ADC 제어 조도센서 (통신, I2C) - WiringPi

```
#include <stdio.h>
#include <wiringPi.h>
#include <wiringPiI2C.h>
int main(void) {
    int fd;
    int prev, a2dVal[4];
    float a2dVol;
    float Vref = 5.0;
    printf("[ADC/DAC Module testing.....]\n");
    if ((fd = wiringPiI2CSetup(0x48)) < 0) {
        printf("wiringPiI2CSetup failed:\n");
    }
    while (1) {
        wiringPiI2CWrite(fd, 0x44);
        prev = wiringPiI2CRead(fd); // Previously byte, garbage
        for(int i = 0; i < 4; i++) a2dVal[i] = wiringPiI2CRead(fd);
        a2dVol = a2dVal[1] * 100.0 / 255;
        printf("VAR = %3.0f %n", a2dVol);
        delay(500);
    }
}
```

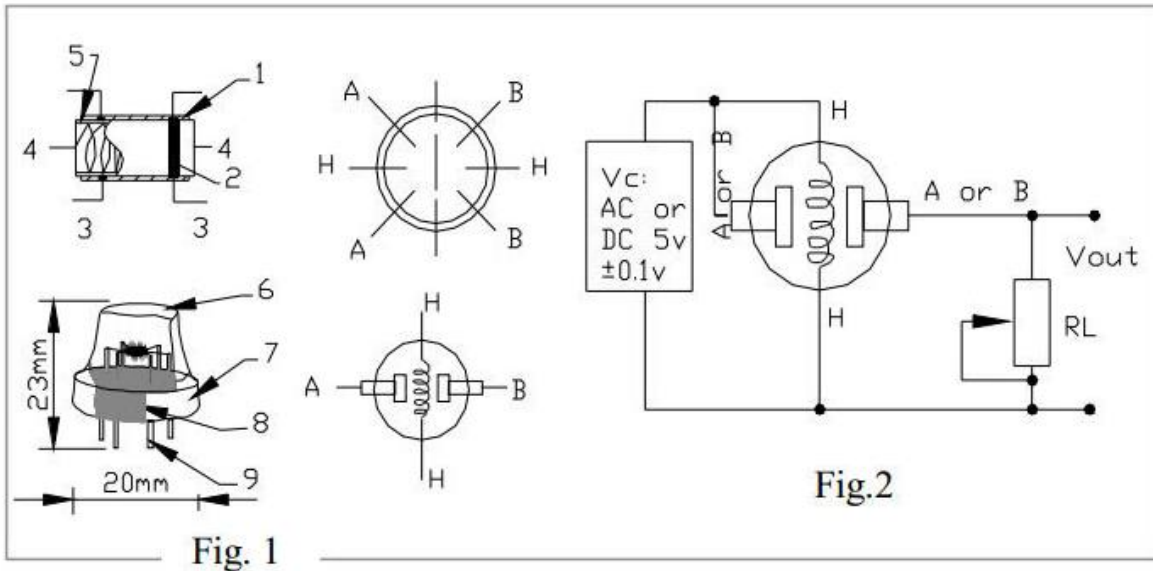


```
bready@raspberrypi:~/AISW $ ./adc_auto
[ADC/DAC Module testing.....]
VAR = 62 %
VAR = 62 %
VAR = 62 %
VAR = 62 %
VAR = 62 %
VAR = 62 %
VAR = 62 %
VAR = 62 %
VAR = 62 %
VAR = 62 %
```


가스센서 제어(통신, I2C)



- 가스 센서는 일산화탄소(CO)를 검출하는 가스 센서다.
- 금속산화물반도체 센서로도 알려져 있으며, 반도체가 가스와 직접 접촉할 때 발생하는 화학반응에 의해 가스를 감지



ADC 제어 가스센서 (통신, I2C) - 파이썬

파이썬 통신 모듈:

```
import smbus
```

```
bus = smbus.SMBus(1)
```

```
# I2C버스가 연결된 smbus 모듈 객체화
```

```
i2c_address = 0x48
```

```
command = 0x44 (데이터 읽어올때)
```

```
# I2C 사용을 위한 모듈 smbus
```

```
import smbus
```

```
# 지연시간 제어를 위해 time 모듈을 사용한다.
```

```
import time
```

```
bus = smbus.SMBus(1)
```

```
i2c_address = 0x48
```

```
# PCF8591 칩에서 데이터를 받기위한 명령어다.
```

```
command = 0x44
```

```
# try-except 는 파이썬의 예외처리 구문으로
```

```
# 키보드로 Ctrl + C를 누를시 프로그램이 종료 된다.
```

```
try:
```

```
    while(1):
```

```
        # i2c의 주소와 명령어를 전송하여 5Byte의 데이터를 읽어온다.
```

```
        # 맨 앞의 dummy data(index 0번 데이터)를 제외하고 뒤 4Byte가 4개의 ADC 포트 데이터이다.
```

```
        adc_data = bus.read_i2c_block_data(i2c_address, command, 5)
```

```
        # I2C로 부터 센서 값을 읽어온다 gas 센서는 2번이다.
```

```
        GAS = adc_data[3]
```

```
        print("GAS : " + str(GAS))
```

```
        time.sleep(0.1)
```

```
# 종료 등의 키보드 인터럽트 발생시 처리 동작
```

```
except KeyboardInterrupt:
```

```
    pass
```

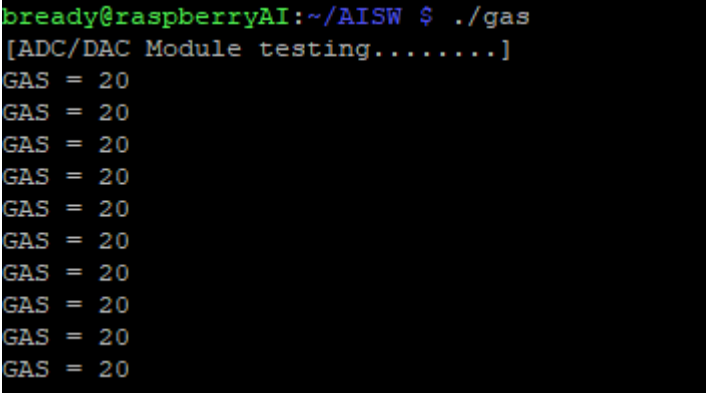


ADC 제어 조도센서 (통신, I2C) - WiringPi

```
#include <stdio.h>
#include <wiringPi.h>
#include <wiringPiI2C.h>
int main(void) {
    int fd;
    int prev, a2dVal[4];

    printf("[ADC/DAC Module testing.....]\n");
    if ((fd = wiringPiI2CSetup(0x48)) < 0) {
        printf("wiringPiI2CSetup failed:\n");
    }

    while (1) {
        wiringPiI2CWrite(fd, 0x44);
        prev = wiringPiI2CRead(fd); // Previously byte, garbage
        for (int i = 0; i < 4; i++) a2dVal[i] = wiringPiI2CRead(fd);
        printf("GAS = %d\n", a2dVal[2]);
        delay(500);
    }
}
```



```
bready@raspberrypi:~/AISW $ ./gas
[ADC/DAC Module testing.....]
GAS = 20
GAS = 20
GAS = 20
GAS = 20
GAS = 20
GAS = 20
GAS = 20
GAS = 20
GAS = 20
GAS = 20
```

PSD센서 제어(통신, I2C)

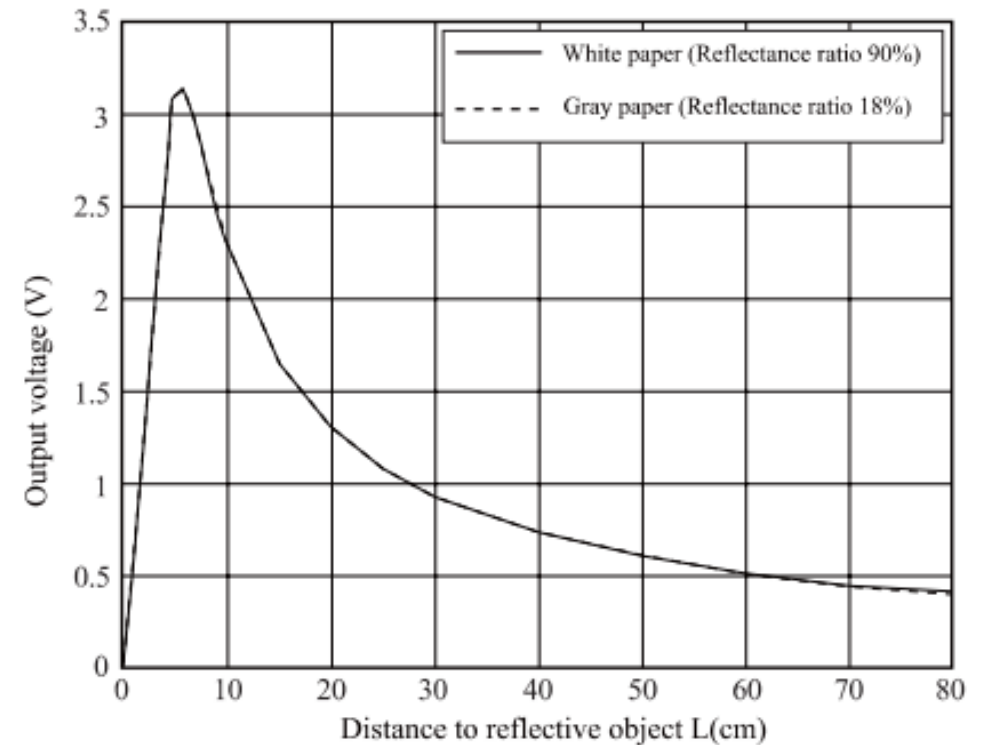
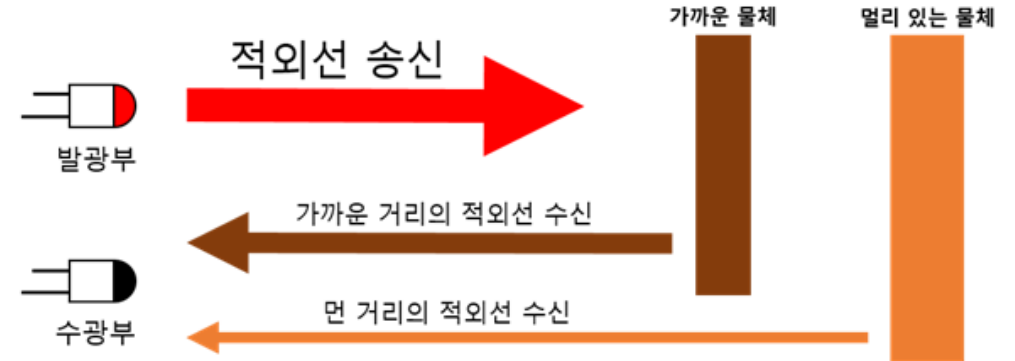


SHARP

GP2Y0A21YK0F

GP2Y0A21YK0F

Distance Measuring Sensor Unit
Measuring distance: 10 to 80 cm
Analog output type



PSD 센서(통신, I2C) - 파이썬

```
# I2C 사용을 위한 모듈 smbus
```

```
import smbus
```

```
# 지연시간 제어를 위해 time 모듈을 사용한다.
```

```
import time
```

```
bus = smbus.SMBus(1)
```

```
i2c_address = 0x48
```

```
# PCF8591 칩에서 데이터를 받기위한 명령어다.
```

```
command = 0x44
```

```
# try-except 는 파이썬의 예외처리 구문으로
```

```
# 키보드로 Ctrl + C를 누를시 프로그램이 종료 된다.
```

```
try:
```

```
    while(1) :
```

```
        # i2c의 주소와 명령어를 전송하여 5Byte의 데이터를 읽어온다.
```

```
        # 맨 앞의 dummy data(index 0번 데이터)를 제외하고 뒤 4Byte가 4개의 ADC 포트 데이터이다.
```

```
        adc_data = bus.read_i2c_block_data(i2c_address, command, 5)
```

```
        # I2C로 부터 센서 값을 읽어온다 PSD 센서는 3번이다.
```

```
        # adc 칩에 입력되는 전압을 계산한 후 => (adc_data[4] / 255.0 * 3.3)
```

```
        # 전압 분배 저항 통과 하기 전의 전압을 계산한다. => * 3 / 2
```

```
        psd_value = (adc_data[4] / 255.0 * 3.3) * 3 / 2
```

```
        psd_value = 29.988 * math.pow(psd_value , -1.173)
```

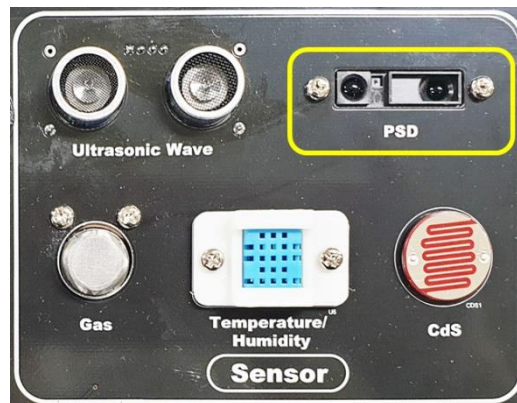
```
        psd_value = round(psd_value, 2)
```

```
        print("PSD : " + str(psd_value) + "cm")        time.sleep(0.1)
```

```
# 종료 등의 키보드 인터럽트 발생시 처리 동작
```

```
except KeyboardInterrupt:
```

```
    pass
```



GP2Y0A21YK

Model: "1080" [10cm to 80cm]

Volt	Distance
2,6	10
2,1	12
1,85	14
1,65	15
1,5	18
1,39	20
1,15	25
0,98	30
0,85	35
0,75	40
0,67	45
0,61	50
0,59	55
0,55	60
0,5	65
0,48	70
0,45	75
0,42	80

Using MS Excel, we can calculate function (For distance > 10cm) :

Distance = 29.988 X POW(Volt , -1.173)

PSD 센서 (통신, I2C) - WiringPi

```
#include <stdio.h>
#include <wiringPi.h>
#include <wiringPiI2C.h>
#include <math.h>
int main(void) {
    int fd;
    int prev, a2dVal[4];
    float psd_value;
    float range;

    printf("[ADC/DAC Module testing.....]\n");
    if ((fd = wiringPiI2CSetup(0x48)) < 0) {
        printf("wiringPiI2CSetup failed:\n");
    }
    while (1) {
        wiringPiI2CWrite(fd, 0x44);
        prev = wiringPiI2CRead(fd); // Previously byte, garbage
        for (int i = 0; i < 4; i++) a2dVal[i] = wiringPiI2CRead(fd);
        psd_value = (a2dVal[3] / 255.0 * 3.3) * 3 / 2;
        range = 29.988 * pow(psd_value, -1.173);
        printf("range = %3.0f cm\n", range);
        delay(50);
    }
}
```

```
bready@raspberrypi:~/AISW $ gcc -o psd_2 psd_2.c -lwiringPi -lm
bready@raspberrypi:~/AISW $
```

-lm 옵션 추가

GP2Y0A21YK

Model: "1080" [10cm to 80cm]

Volt	Distance
2,6	10
2,1	12
1,85	14
1,65	15
1,5	18
1,39	20
1,15	25
0,98	30
0,85	35
0,75	40
0,67	45
0,61	50
0,59	55
0,55	60
0,5	65
0,48	70
0,45	75
0,42	80

Using MS Excel, we can calculate function (For distance > 10cm) :

Distance = 29.988 X POW(Volt , -1.173)

PSD 센서 (통신, I2C) - WiringPi

```
#include <stdio.h>
#include <wiringPi.h>
#include <wiringPiI2C.h>
```

```
int main(void) {
    int fd;
    int prev, a2dVal[4];
    float psd_value;
    float range;

    printf("[ADC/DAC Module testing.....]\n");
    if ((fd = wiringPiI2CSetup(0x48)) < 0) {
        printf("wiringPiI2CSetup failed:\n");
    }

    while (1) {
        wiringPiI2CWrite(fd, 0x44);
        prev = wiringPiI2CRead(fd); // Previously byte, garbage
        for (int i = 0; i < 4; i++) a2dVal[i] = wiringPiI2CRead(fd);
        psd_value = (a2dVal[3] / 255.0 * 3.3) * 3 / 2;
        range = 19.8 / (psd_value - 0.228);
        printf("range = %3.0f cm\n", range);
        delay(50);
    }
}
```

$u = f(1/d)$ may be approximated by a straight line $u = mx + b$ with $m = 19.8$ and $b = 0.228$ in the range of $d = 7$ to $d = 80$ cm.

